

# Quantum correction with three codes

<sup>1</sup>Aziz Mouzali, <sup>2</sup>Fatiha Merazka

<sup>1</sup>Faculty of Sciences, University of Blida, Algérie.

<sup>2</sup>*Electronic* & Computer engineering Faculty, University of Science & Technology  
Houari Boumediene, Algeria. fmerazka@usthb.dz

## Abstract

In this paper, we provide an implementation of five, seven and nine-qubits error correcting codes on a classical computer using the quantum simulator Feynman program. We also compare the three codes by computing the fidelity when double errors occurs in a depolarizing channel. As triple errors and more are considered very unlikely, it has negligible effect on the next results.

**Keywords:** Quantum error correction, qubits, Feynman program, Maple, Fidelity.

## 1 Introduction

The classical information processing is performed with physical supports, which are governed by the classical physics laws. The quantum information processing uses particles, which obeys to quantum mechanics (electrons, ions, photons...). An electron can carry information : according to its spin, it represents the bit 0 or the bit 1. The quantum information support called qubit is a physical system, which can be in two quantum states representing the bits 0 or 1 [1]. The advantage of quantum information processing is the possibility for the qubit to be in a two states superposition, which is forbidden for the classical bit. However, any environment qubit interaction will break this superposition at one of its two possible states. Another essential tool is the entanglement between the qubits, which is used in quantum computing. An n-qubits system entangled state is not the n-tensor product of every qubit state, as it is the case for a separable state. Unfortunately, the entanglement is very sensitive to the environment noise. The superposition and entanglement loss is called decoherence and introduces errors in computation, which requires quantum error correcting code. In fact, while the classical computation error probability is negligible, the error probability is very high in quantum computation [2]. The difference comes from the physical nature of the information support : the electrons number in a capacitor can vary without changing the bit, but if an electron spin changes then the bit value will change. As the experimental protection against decoherence is very hard, several quantum codes have been built to correct computation errors. A quantum error correcting code aims to identify the qubits states alteration caused by the noises and

then to recover the correct states. A quantum code is similar to a classical code, but has one particularity: We cannot copy the bit to protect it because it is impossible to clone the quantum information. Also, we can not read out directly the information contained in the useful qubit. In fact, we intricate the superposed qubit state carrying the useful information with a number of additional qubits states, called ancillas, to build the codedwords. If an errors affects the qubits, a correcting procedure allows getting a system separable state, containing the useful information [1].

We propose in this paper, to simulate the five, seven and nine qubits code This paper is organized as follows. In section 2, we take a look at some basics of quantum error correcting codes, bit flip and phase flip correction and error measurement . In section 3, we describe the bit flip, phase flip and the Shor's nine qubits codes. Sections 4, 5 and 6 are dedicated to the correction of double errors respectively by the nine, seven and five qubits codes. Section 7 is devoted to the definition of fidelity used in this work, then in Section 8, we present and discuss the experimental results of this fidelity when double errors occur in depolarizing channel and corrected by the five, seven and nine qubits codes. Finally, we offer our conclusions in Section 9.

## 2 Basics of Quantum Error Correction

Consider an n-qubits system in interaction with the environment. We represent an error as unitary transformation U over all the "n-qubits+environment" state. This error can be a bit flip X, a phase flip Z and both of them  $Y=iXZ$ . We give below, the matricial representation of these errors and their effect on a one superposed state  $\alpha|0\rangle + \beta|1\rangle$ . The Kets  $|0\rangle$  and  $|1\rangle$  correspondent, for example, to respectively the spin  $+1/2$  and  $-1/2$  of an electron , or the fundamental and excited states of an atom. The factors  $\alpha$  and  $\beta$  are in general complex number and give the probability  $|\alpha|^2$  and  $|\beta|^2$  ( $|\alpha|^2 + |\beta|^2 = 1$ ) that the qubit is respectively in the states  $|0\rangle$  and  $|1\rangle$ .

If none error occurs, we use the identity matrix I:

$$I(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle + \beta|1\rangle \quad \text{or} \quad I \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

The Bit flip error is represented by the X Pauli matrix :

$$X(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle \quad \text{or} \quad X \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

The phase flip error is represented by the Z Pauli matrix :

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle \quad \text{or} \quad Z \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}$$

The Bit and phase flip error is represented by the  $Y=-iXZ$  Pauli matrix :

$$Y(\alpha|0\rangle + \beta|1\rangle) = i(-\beta|0\rangle + \alpha|1\rangle) \text{ or } Y \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = i \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = i \begin{pmatrix} -\beta \\ \alpha \end{pmatrix}$$

We also use in quantum error correction the Hadamard (H), controlled-not (CNOT or CN) and Toffoli (T) gates. They act respectively on one, two and three qubits. We give below their matricial representation and effect on some states.

$$\text{Hadamard gate : } H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

This gate transforms a simple state in a superposed state and inversely:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle; \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle;$$

$$H|+\rangle = |0\rangle; \quad H|-\rangle = |1\rangle$$

$$\text{Controlled-not gate (CNOT) : } CN = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

This gate acting on two qubits, flips the second one (target) qubit only if the first one (controller) is equal to 1:

$$CN|00\rangle = |00\rangle \quad ; \quad CN|01\rangle = |01\rangle \quad ; \quad CN|10\rangle = |11\rangle \quad ; \quad CN|11\rangle = |10\rangle$$

$$\text{Toffoli gate (CCN) : } T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

This gate acting on three qubits, flips the third one (target) only if the two others are equal to 1:

$$T|110\rangle = |111\rangle \quad ; \quad T|111\rangle = |110\rangle$$

### 3 Quantum error correcting codes

We present below some quantum error correcting codes (QECC) used to protect useful information stored in a superposed qubit state  $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ .

#### 3.1 Bit flip code

To correct a bit flip, we use a redundant code by coding  $|0\rangle$  as  $|000\rangle$  and  $|1\rangle$  as  $|111\rangle$ . The initial state is coded as  $\alpha|000\rangle + \beta|111\rangle$  and becomes after error on, for example, the third qubit  $\alpha|001\rangle + \beta|110\rangle$ . We have supposed that a bit flip occurs on all the superposition terms. To identify the affected qubit we add a fourth one initialized to  $|\Psi_4\rangle = |0\rangle$  which will locate and record its position. The system state will be then  $(\alpha|001\rangle + \beta|110\rangle)|0\rangle$ . We determine by calculus the error position and obtain the state  $(\alpha|001\rangle + \beta|110\rangle)|\Psi_4^3\rangle$  where the fourth qubit state  $|\Psi_4^3\rangle$  indicates that the third qubit is affected by an error. After correcting the designed qubit, the global state becomes  $(\alpha|000\rangle + \beta|111\rangle)|\Psi_4^3\rangle$ . Finally, we suppress the redundancy and return to the initial state  $\alpha|0\rangle + \beta|1\rangle$ . This procedure is also valid when the error occurs on superposition. Consider a noise which produces the state  $(\alpha|100\rangle + \beta|011\rangle + \alpha|001\rangle + \beta|110\rangle)|0\rangle$ . Adding the error register and locating the affected qubit gives  $\frac{1}{\sqrt{2}}[(\alpha|100\rangle + \beta|011\rangle)|1\rangle + (\alpha|001\rangle + \beta|110\rangle)|\Psi_4^{13}\rangle]$  where the state  $|\Psi_4^{13}\rangle$  indicates that the first and the third qubits are affected by errors. Measuring the error register gives  $(\alpha|100\rangle + \beta|011\rangle)|\Psi_4^1\rangle$  or  $(\alpha|001\rangle + \beta|110\rangle)|\Psi_4^3\rangle$ . Finally, we flip the qubits designed by the error register to obtain the state  $\alpha|000\rangle + \beta|111\rangle$ . Then, we suppress the redundancy to return to the original state  $\alpha|0\rangle + \beta|1\rangle$  [2-4].

Figure 1 shows the Bit flip code circuit. The encoding and decoding circuits are located in the time intervals  $[t_0, t_1]$  and  $[t_2, t_3]$ , respectively. The useful information is stored on the first qubit state  $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ . The second and third qubits are added to recover this state if a bit flip occurs in  $[t_1, t_2]$  [2-4].

At time  $t_0$ , the initial separable state system is :

$$|\Psi_0\rangle = |\Psi\rangle|0\rangle|0\rangle = \alpha|000\rangle + \beta|100\rangle \quad (1)$$

After applying the coding and decoding circuit we obtain at time  $t_3$ , the disintricated state:

$$|\Psi_3\rangle = \alpha|011\rangle + \beta|111\rangle = (\alpha|0\rangle + \beta|1\rangle)|11\rangle \quad (2)$$

We have then, recovered the first qubit state, which contains the useful information. The bit flip in the other two qubits indicates, that a bit flip has affected the

first one. If a bit flip occurs on the second and third qubits, we obtain the states, respectively as:

$$|\Psi_2\rangle = \alpha |010\rangle + \beta |101\rangle; |\Psi'_2\rangle = \alpha |010\rangle + \beta |110\rangle; |\Psi_3\rangle = \alpha |0\rangle + \beta |1\rangle |10\rangle \quad (3)$$

$$|\Psi_2\rangle = \alpha |001\rangle + \beta |110\rangle; |\Psi'_2\rangle = \alpha |001\rangle + \beta |101\rangle; |\Psi_3\rangle = \alpha |0\rangle + \beta |1\rangle |01\rangle \quad (4)$$

Appendix B shows the bit flip code simulation made with the Feynman program. The procedure called "Bitflip" takes as an input the bit flip affected qubit number n=1, 2 or 3. The input n=0 corresponds to the case where no errors occur.

### 3.2 Phase flip code

We want to protect the qubit state  $\alpha |0\rangle + \beta |1\rangle$  from a phase flip error. The idea is to transform a phase flip, which is uncorrectable, into a bit flip, which is correctable. This state is then coded as  $\alpha |+++\rangle + \beta |--\rangle$  where  $|+\rangle$  and  $|-\rangle$  are obtained by the Hadamard gate:  $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$  and  $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$ . A phase flip on the first qubit gives the state  $\alpha |-++\rangle + \beta |+--\rangle$ . The correction begins by a three transformation  $H^{\otimes 3}$  which allows to return to the basic state  $H^{\otimes 3}\alpha |-++\rangle + \beta |+--\rangle = \alpha |100\rangle + \beta |011\rangle$ . Then we correct the affected qubit to restore the state  $\alpha |000\rangle + \beta |111\rangle$ . Finally, we apply  $H^{\otimes 3}$  to return at the coded state  $\alpha |+++\rangle + \beta |--\rangle$  [2-4].

Figure 2 shows the phase flip code circuit. The useful information is stored on the first qubit state  $|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle$ . The encoding and decoding circuits are located in the time intervals  $[t_0, t_1]$  and  $[t_2, t_3]$ , respectively. This code uses three qubits to correct a phase flip, that affect one of them between  $t_1$  and  $t_2$  [2-4].

After applying the coding and decoding circuit we obtain at time  $t_3$ , the disintricated state :

$$|\Psi_3\rangle = \frac{1}{\sqrt{2}} [\alpha |0\rangle + \beta |1\rangle] |11\rangle = |\Psi\rangle |11\rangle \quad (5)$$

We have then, recovered the first qubit state, which contains the useful information. The bit flip in the other two qubits indicates, that a phase flip has affected the first one.

Appendix B shows the phase flip code simulation made with the Feynman program. The procedure called "Bitflip" takes as an input the phase flip affected qubit number n=1, 2 or 3. The input n=0 corresponds to the case where no errors occur.

We give, below, the outputs for the fourth n values. The running time is about two seconds.

### 3.3 Shor code

The Shor code uses nine qubits to correct a combined phase and bit flip error on one qubit. We describe below the encoding, correction and decoding procedures of this code.

#### 3.3.1 Encoding

The initial state  $\alpha|0\rangle + \beta|1\rangle$  is coded  $\alpha|+++\rangle + \beta|---\rangle = \alpha\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes 3} + \beta\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right)^{\otimes 3}$  so it could be protected against phase flip. Then we protect against bit flip by coding each  $|0\rangle$  as  $|000\rangle$  and each  $|1\rangle$  as  $|111\rangle$  and obtain :  $\alpha\left(\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)\right)^{\otimes 3} + \beta\left(\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)\right)^{\otimes 3}$ . The Shor algorithm, which will be described below, corrects first the bit flip, then the phase flip [2][3][4].

Figure 3 shows the circuit of this code [4]. This code uses nine qubits to correct a single X, Y or Z error on only one of them. It is the concatenation of the two codes described above: the bit and the phase flip codes [4].

The useful information is stored on the the first qubit state  $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ . The others ones, are added to recover this state if an error occurs between the instants  $t_1$  and  $t_2$ . We can remove or use them again , after recovering their initial state  $|0\rangle$ . Boxes  $C_B$  and  $D_B$  are called coding and decoding boxes, respectively. They correspond to the circuits shown on Figure 4. Symbol D indicates the error detection measured on the six additional qubits :

After applying the coding circuit, we obtain at time  $t_1$  the coded state:

$$|\Psi_1\rangle = \frac{1}{2^{3/2}} [\alpha(|000\rangle + |111\rangle)^{\otimes 3} + \beta(|000\rangle - |111\rangle)^{\otimes 3}] \quad (6)$$

or 
$$|\Psi_1\rangle = [\alpha|0_L\rangle + \beta|1_L\rangle] \quad (7)$$

The label 'L' indicates the logical qubit state, which is different from the physical qubit state  $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ . The codewords of this nine qubits code are:

$$|0_L\rangle = \frac{1}{2^{3/2}}(|000\rangle + |111\rangle)^{\otimes 3} \quad \text{and} \quad |1_L\rangle = \frac{1}{2^{3/2}}(|000\rangle - |111\rangle)^{\otimes 3} \quad (8)$$

#### 3.3.2 X error on the useful qubit

If an a bit flip X occurs between  $t_1$  and  $t_2$  the affected decodedstate obtained at time  $t_3$  is :

$$|\Psi_3\rangle = (\alpha|000\rangle + \beta|100\rangle)|11\rangle(|00\rangle)^{\otimes 2} = |\Psi\rangle|00\rangle|11\rangle(|00\rangle)^{\otimes 2} \quad (9)$$

This final state is separable and the first qubit initial state, carrying the useful information has been recovered. The measured modification on the forth and fifth states qubits lets to know that an error occurred. The additional qubits can now, be removed or used again, after putting them on the  $|0\rangle$  state.

### 3.3.3 Z error on the useful qubit

If an a bit flip X occurs between  $t_1$  and  $t_2$  the affected decodedstate obtained at time  $t_3$  is :

$$|\Psi_3\rangle = |\Psi\rangle|11\rangle(|00\rangle)^{\otimes 3} \quad (10)$$

The first qubit state is then recovered. The modification on the second and third qubits states, indicates an error occurrence.

### 3.3.4 Y error on the useful qubit

If an a bit flip X occurs between  $t_1$  and  $t_2$  the affected decodedstate obtained at time  $t_3$  is :

$$|\Psi_3\rangle = i|\Psi\rangle|1111\rangle(|00\rangle)^{\otimes 2} \quad (11)$$

The first qubit state is then recovered and the measured modification on the second, third, forth and fifth qubit states indicates that an error has occurred. We remark, that the states  $|\Psi\rangle$  and  $i|\Psi\rangle$  are physically the same, so we do not need to suppress the complex number  $i$ .

### 3.3.5 Arbitrary error on the useful qubit

Suppose now that an arbitrary error  $E = c_o I + c_x X + c_y Y + c_z Z$  occurs on the first qubit so that  $|\Psi_2\rangle = E|\Psi_1\rangle$ . The coefficients  $c_x$ ,  $c_y$  and  $c_z$  give respectively, the probability  $|c_x|^2$ ,  $|c_y|^2$  and  $|c_z|^2$  that an X, Y or Z error affects the qubit. The same calculus allows to obtain the state  $|\Psi_3\rangle$  as a linear combination of the correct state  $|\Psi_3\rangle = |\Psi\rangle(|00\rangle)^{\otimes 4}$  and the three states  $|\Psi_3\rangle$  calculated for respectively, the simple X, Y or Z errors :

$$|\Psi_3\rangle = |\Psi\rangle[c_o|0000\rangle + c_x|0011\rangle + ic_y|1111\rangle + c_z|1100\rangle](|00\rangle)^{\otimes 2} \quad (12)$$

### 3.3.6 Detection and correction error before decoding

To detect an error in the state  $|\Psi_2\rangle$ , we need to locate the affected qubit state. We then correct it by making detection measure before the decoding procedure. We use for detection the stabilizer group  $G = \{g_i, i = 1..8\}$  of the Shor code which is:  $\{Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\}$ . Every generator  $g_i$  verify the relation  $g_i |\Psi_1\rangle = |\Psi_1\rangle$  with  $|\Psi_1\rangle = [\alpha |0_L\rangle + \beta |1_L\rangle]$ .

We begin by applying all the six first generators on  $|\Psi_2\rangle$  to verify that no bit flip occurred. We use for this, the fact that if no error occurs ( $|\Psi_2\rangle = |\Psi_1\rangle$ ) or if only a phase flip occurs in  $|\Psi_2\rangle$  all the terms in the superposition are  $\pm |000\rangle$  or  $\pm |111\rangle$ , so that we obtain:

$$Z_1Z_2 |\Psi_2\rangle = Z_2Z_3 |\Psi_2\rangle = Z_4Z_5 |\Psi_2\rangle = Z_5Z_6 |\Psi_2\rangle = Z_7Z_8 |\Psi_2\rangle = Z_8Z_9 |\Psi_2\rangle = |\Psi_2\rangle \quad (13)$$

For example, if an X error occurs on the first qubit, we obtain :

$$|\Psi_2\rangle = \frac{1}{2^{3/2}} [\alpha(|100\rangle + |011\rangle)(|000\rangle + |111\rangle)^{\otimes 2} + \beta(|100\rangle - |011\rangle)(|000\rangle - |111\rangle)^{\otimes 2}] \quad (14)$$

$$Z_1Z_2 |\Psi_2\rangle = -|\Psi_2\rangle \quad \text{and} \quad Z_2Z_3 |\Psi_2\rangle = |\Psi_2\rangle \quad (15)$$

The results are depicted on the table 1, where the first column gives the bit flip affected qubit number. The values +1 and -1 are the  $Z_iZ_{i+1}$  operators eigenvalues when applied on the  $|\Psi_2\rangle$  state. The last column gives the correction recovering the original state  $|\Psi_1\rangle$ .

Qubit	$Z_1Z_2$	$Z_2Z_3$	$Z_4Z_5$	$Z_5Z_6$	$Z_7Z_8$	$Z_8Z_9$	Correction
1	-1	+1	+1	+1	+1	+1	$X_1$
2	-1	-1	+1	+1	+1	+1	$X_2$
3	+1	-1	+1	+1	+1	+1	$X_3$
4	+1	+1	-1	+1	+1	+1	$X_4$
5	+1	+1	-1	-1	+1	+1	$X_5$
6	+1	+1	+1	-1	+1	+1	$X_6$
7	+1	+1	+1	+1	-1	+1	$X_7$
8	+1	+1	+1	+1	-1	-1	$X_8$
9	+1	+1	+1	+1	+1	-1	$X_9$

**Table 1: Action of the generators on a bit flip affected qubit.**

The obtained eigenvalues allow to identify the affected qubit 'i' and then to correct it by applying the  $X_i$  operator. In the second step, we detect the phase flip by applying on  $|\Psi_2\rangle$  the two generators  $g_7 = X_1X_2X_3X_4X_5X_6$  and  $g_8 = X_4X_5X_6X_7X_8X_9$ .

For example, If a phase flip occurs on the first qubit we obtain:



$$|\Psi_2\rangle = \frac{1}{2^{3/2}} [\alpha(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)^{\otimes 2} + \beta(|000\rangle + |111\rangle)(|000\rangle - |111\rangle)^{\otimes 2}] \quad (17)$$

$$g_7 |\Psi_2\rangle = -|\Psi_2\rangle \quad \text{and} \quad g_8 |\Psi_2\rangle = |\Psi_2\rangle \quad (18)$$

The results are summarized on the table 2, where the first column gives the phase flip affected qubit number. The values +1 and -1 are the eigenvalues of the operators  $g_7$  and  $g_8$  when applied on the  $|\Psi_2\rangle$  state. The last column gives the correction, which recovers the original state  $|\Psi_1\rangle$ . We note that this correction procedure works only if a single error occurs and does not work in the case of an arbitrary error.

Qubit	$g_7$	$g_8$	Correction
1	-1	+1	$Z_1 Z_2 Z_3$
2	-1	+1	$Z_1 Z_2 Z_3$
3	-1	+1	$Z_1 Z_2 Z_3$
4	-1	-1	$Z_4 Z_5 Z_6$
5	-1	-1	$Z_4 Z_5 Z_6$
6	-1	-1	$Z_4 Z_5 Z_6$
7	+1	-1	$Z_7 Z_8 Z_9$
8	+1	-1	$Z_7 Z_8 Z_9$
9	+1	-1	$Z_7 Z_8 Z_9$

**Table 2: The generators's action on a phase flip affected qubit.**

### 3.3.7 Simulation results

The Feynman program described in [5][6][7][8] is a set of procedures supporting the definition and manipulation of the states of an n-qubits system and the unitary gates acting on them. We have simulated the Shor code on Maple 11 using Feynman program as library. The appendix D contains this code with some included explanations. As the algorithm structure aims to reduce the running time, we avoided the several matrix product, because it takes too much time. It contains four parts: coding, error, correction and decoding, giving respectively, the states Psi1, Psi2, Psi2b, and Psi3.

The table 3 gives the simulation results when no correction is made before or after decoding. The first column contains the  $X_i$ ,  $Z_i$  and  $Y_i$  input errors on the qubit "i" and the second one the obtained output error. We note that, the first qubit state is always recovered without correction, at the end of the quantum circuit. Therefore, we can always suppress the ancillas without affecting the useful information.

Also, we also note that, apart  $X_8$  and  $X_9$ , all the other  $X_i$  input errors are corrected at the output. We remark that, all the  $Z_i$  input errors are transformed in  $X_i$  output errors by the decoding circuit. Moreover, the number of affected qubits at the output

is one or two for the  $X_i$  and  $Z_i$  input errors and two, three or four for the  $Y_i$  input errors. Finally, we note that every  $Y_i$  input error gives a distinct output error, which then allows to identify clearly this input error. Inversely, the same output errors can be produced by different  $X_i$  or  $Z_i$  input errors.

Input	Output	Input	Output	Input	Output
$X_1$	$X_4X_5$	$Z_1$	$X_2X_3$	$Y_1$	$-iX_2X_3X_4X_5$
$X_2$	$X_4$	$Z_2$	$X_2X_3$	$Y_2$	$-iX_2X_3X_4$
$X_3$	$X_5$	$Z_3$	$X_2X_3$	$Y_3$	$-iX_2X_3X_5$
$X_4$	$X_6X_7$	$Z_4$	$X_2$	$Y_4$	$-iX_2X_6X_7$
$X_5$	$X_6$	$Z_5$	$X_2$	$Y_5$	$-iX_2X_6$
$X_6$	$X_7$	$Z_6$	$X_2$	$Y_6$	$-iX_2X_7$
$X_7$	$X_8X_9$	$Z_7$	$X_3$	$Y_7$	$-iX_3X_8X_9$
$X_8$	$X_8$	$Z_8$	$X_3$	$Y_8$	$-iX_3X_8$
$X_9$	$X_9$	$Z_9$	$X_3$	$Y_9$	$-iX_3X_9$

**Table 3:** The output errors for a bit and phase flip on one qubit.

## 4 Correction of double errors

Consider now double errors occurring before decoding and corrected by single error having same syndrome. Tables 5 give for each input error  $E$ , the syndromes  $S$  and error  $E_i$  affecting the protected qubit  $i=1$  after correction and decoding.

$E$	$S$	$E_i$	$E$	$S$	$E_i$
$X_1, X_2X_3$	10000000	$I_i, Z_i$	$Y_1, X_1Z_{2,3}$	10000010	$I_i$
$X_2, X_1X_3$	11000000	$I_i, Z_i$	$Y_2, X_2Z_{1,3}$	11000010	$I_i$
$X_3, X_1X_2$	01000000	$I_i, Z_i$	$Y_3, X_3Z_{1,2}$	01000010	$I_i$
$X_4, X_5X_6$	00100000	$I_i, Z_i$	$Y_4, X_4Z_{5,6}$	00100011	$I_i$
$X_5, X_4X_6$	00110000	$I_i, Z_i$	$Y_5, X_5Z_{4,6}$	00110011	$I_i$
$X_6, X_4X_5$	00010000	$I_i, Z_i$	$Y_6, X_6Z_{4,5}$	00010011	$I_i$
$X_7, X_8X_9$	00001000	$I_i, Z_i$	$Y_7, X_7Z_{8,9}$	00001001	$I_i$
$X_8, X_7X_9$	00001100	$I_i, Z_i$	$Y_8, X_8Z_{7,9}$	00001101	$I_i$
$X_9, X_7X_8$	00000100	$I_i, Z_i$	$Y_9, X_9Z_{7,8}$	00000101	$I_i$

**Table 4a :** Syndromes  $S$  and the error  $E_i$  affecting the protected qubits " $i = 1$ " after correction by an operators  $X_k$  or  $Y_k$  ( $k = 1..9$ ).

Errors	$S$	$E_i$
$(Z_1, Z_2, Z_3), (Z_4Z_{7,8,9}, Z_5Z_{7,8,9}, Z_6Z_{7,8,9})$	00000010	$(I_i), (X_i)$
$(Z_4, Z_5, Z_6), (Z_1Z_{7,8,9}, Z_2Z_{7,8,9}, Z_3Z_{7,8,9})$	00000011	$(I_i), (X_i)$
$(Z_7, Z_8, Z_9), (Z_1Z_{4,5,6}, Z_2Z_{4,5,6}, Z_3Z_{4,5,6})$	00000001	$(I_i), (X_i)$
$(Z_1Z_{2,3}, Z_2Z_3, Z_4Z_{5,6}, Z_5Z_6, Z_7Z_{8,9}, Z_8Z_9)$	00000000	$(I_i)$

**Table 4b:** Syndromes  $S$  and error  $E_i$  on the protected qubits " $i = 1$ " after correction by an operator  $Z_k$ .

$E$	$S$	$E$	$S$	$E$	$S$
$X_1 X_4$	10100000	$X_2 X_7$	11001000	$X_4 X_7$	00101000
$X_1 X_5$	10110000	$X_2 X_8$	11001100	$X_4 X_8$	00101100
$X_1 X_6$	10010000	$X_2 X_9$	11000100	$X_4 X_9$	00100100
$X_1 X_7$	10001000	$X_3 X_4$	01100000	$X_5 X_7$	00111000
$X_1 X_8$	10001100	$X_3 X_5$	01110000	$X_5 X_8$	00111100
$X_1 X_9$	10000100	$X_3 X_6$	01010000	$X_5 X_9$	00110100
$X_2 X_4$	11100000	$X_3 X_7$	01001000	$X_6 X_7$	00011000
$X_2 X_5$	11110000	$X_3 X_8$	01001100	$X_6 X_8$	00011100
$X_2 X_6$	11010000	$X_3 X_9$	01000100	$X_6 X_9$	00010100

**Table 4c :** Syndromes  $S$  of double channels errors  $X_k X_l$  not affecting the protected qubits after correction.

$E$	$S$	$E$	$S$
$X_1 Z_{4,5,6}$	10000011	$X_4 Z_{1,2,3}$	00100010
$X_1 Z_{7,8,9}$	10000001	$X_5 Z_{1,2,3}$	00110010
$X_2 Z_{7,8,9}$	11000001	$X_6 Z_{1,2,3}$	00010010
$X_2 Z_{4,5,6}$	11000011	$X_7 Z_{1,2,3}$	00001010
$X_3 Z_{4,5,6}$	01000011	$X_8 Z_{1,2,3}$	00001110
$X_3 Z_{7,8,9}$	01000001	$X_9 Z_{1,2,3}$	00000110
$X_4 Z_{7,8,9}$	00100001	$X_7 Z_{4,5,6}$	00001011
$X_5 Z_{7,8,9}$	00110001	$X_8 Z_{4,5,6}$	00001111
$X_6 Z_{7,8,9}$	00010001	$X_9 Z_{4,5,6}$	00000111

**Table 4d:** Syndromes of double channels errors  $X_k Z_l$  not affecting the protected qubits after correction.

## 5 The seven qubits code

This code described in [9][10], uses seven qubits to protect one of them in a superposed state from any error  $X$ ,  $Y$  or  $Z$ . The tables 6 give syndromes and error  $E_i$  affecting the protected qubits " $i = 1$ " (after correction) for different single and double channels errors. The generators of this code are :  $g_1 = X_4 X_5 X_6 X_7$ ,  $g_2 = X_2 X_3 X_6 X_7$ ,  $g_3 = X_1 X_3 X_5 X_7$ ,  $g_4 = Z_4 Z_5 Z_6 Z_7$ ,  $g_5 = Z_2 Z_3 Z_6 Z_7$ ,  $g_6 = Z_1 Z_3 Z_5 Z_7$ .

$E$	$S$	$E_i$	$E$	$S$	$E_i$
$X_1, (X_2 X_3, X_4 X_5, X_6 X_7)$	000001	$I_i, (X_i)$	$Y_1$	001001	$I_i$
$X_2, (X_1 X_3, X_4 X_6, X_5 X_7)$	000010	$I_i, (X_i)$	$Y_2$	010010	$I_i$
$X_3, (X_1 X_2, X_5 X_6, X_4 X_7)$	000011	$I_i, (X_i)$	$Y_3$	011011	$I_i$
$X_4, (X_1 X_5, X_2 X_6, X_3 X_7)$	000100	$I_i, (X_i)$	$Y_4$	100100	$I_i$
$X_5, (X_1 X_4, X_2 X_7, X_3 X_6)$	000101	$I_i, (X_i)$	$Y_5$	101101	$I_i$
$X_6, (X_1 X_7, X_2 X_4, X_3 X_5)$	000110	$I_i, (X_i)$	$Y_6$	110110	$I_i$
$X_7, (X_1 X_6, X_2 X_5, X_3 X_4)$	000111	$I_i, (X_i)$	$Y_7$	111111	$I_i$

**Table 5a:** Syndromes and error  $E_i$  on the protected qubits "i" after correction by an operators  $X_k$  or  $Y_k$ .

<i>Error</i>	<i>S</i>	<i>E<sub>i</sub></i>
$Z_1, (Z_6 Z_7, Z_2 Z_3, Z_4 Z_5)$	001000	$I_i, (Z_i)$
$Z_2, (Z_1 Z_3, Z_4 Z_6, Z_5 Z_7)$	010000	$I_i, (Z_i)$
$Z_3, (Z_4 Z_7, Z_1 Z_2, Z_5 Z_6)$	011000	$I_i, (Z_i)$
$Z_4, (Z_3 Z_7, Z_1 Z_5, Z_2 Z_6)$	100000	$I_i, (Z_i)$
$Z_5, (Z_2 Z_7, Z_1 Z_4, Z_3 Z_6)$	101000	$I_i, (Z_i)$
$Z_6, (Z_1 Z_7, Z_2 Z_4, Z_3 Z_5)$	110000	$I_i, (Z_i)$
$Z_7, (Z_1 Z_6, Z_2 Z_5, Z_3 Z_4)$	111000	$I_i, (Z_i)$

**Table 5b :** Syndromes and the error affecting the protected qubits after correction by an operator  $Z_k$ .

<i>Errors</i>	<i>S</i>	<i>Errors</i>	<i>S</i>	<i>Errors</i>	<i>S</i>
$X_1 Z_4$	100001	$Z_1 X_4$	001100	$Z_1 X_3$	001011
$X_1 Z_5$	101001	$Z_2 X_4$	010100	$X_1 Z_2$	010001
$X_1 Z_6$	110001	$Z_3 X_4$	011100	$X_1 Z_3$	011001
$X_1 Z_7$	111001	$Z_1 X_5$	001101	$X_2 Z_1$	001010
$X_2 Z_7$	111010	$Z_2 X_5$	010101	$X_2 Z_3$	011010
$X_3 Z_4$	100011	$Z_1 X_6$	001110	$X_2 Z_4$	100010
$X_3 Z_6$	110011	$Z_2 X_6$	010110	$X_2 Z_5$	101010
$X_3 Z_7$	111011	$Z_3 X_6$	011110	$X_2 Z_6$	110010
$X_4 Z_7$	111100	$Z_1 X_7$	001111	$Z_2 X_3$	010011
$X_5 Z_7$	111101	$Z_2 X_7$	010111	$X_4 Z_5$	101100
$X_6 Z_5$	101110	$Z_3 X_7$	011111	$X_4 Z_6$	110100
$X_6 Z_7$	111110	$Z_4 X_5$	100101	$Z_4 X_6$	100110
$X_5 Z_6$	110101	$Z_3 X_5$	011101	$X_3 Z_5$	101011

**Table 5c:** Syndromes of double channels errors  $Z_k X_l$  not affecting the protected qubits after correction.

## 6 The five qubits code

This code is described in [11] and uses five qubits to protect one of them in a superposed state from any error X, Y or Z. The four stabilizers of this code are:

$$g_1 = XXZIZ \quad g_2 = ZXXZI \quad g_3 = IZXXZ \quad g_4 = ZIZXX$$

The tables 7 give the syndromes S of single and double errors occurring in the transmitting channel. The double errors are corrected as the single error having same syndrome. The third column gives the error  $E_i$  affecting after decoding the first to be protected qubit "i = 1"  $|\Psi_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle$ .

Error	S	E <sub>i</sub>
$X_i, (Z_3Z_4), (X_4Z_5, Z_2X_3)$	0101	$I_i, (X_i), (Z_i)$
$X_2, (Z_4Z_5), (Z_iX_5, Z_3X_4)$	0010	$I_i, (X_i), (Z_i)$
$X_3, (Z_iZ_5), (X_iZ_2, Z_4X_5)$	1001	$I_i, (X_i), (Z_i)$
$X_4, (Z_iZ_2), (X_iZ_5, X_2Z_3)$	0100	$I_i, (X_i), (Z_i)$
$X_5, (Z_2Z_3), (X_3Z_4, Z_iX_2)$	1010	$I_i, (X_i), (Z_i)$
$Z_i, (X_2X_5), (X_3Z_5, Z_2X_4)$	1000	$I_i, (X_i), (Z_i)$
$Z_2, (X_iX_3), (Z_iX_4, Z_3X_5)$	1100	$I_i, (X_i), (Z_i)$
$Z_3, (X_2X_4), (X_iZ_4, Z_2X_5)$	0110	$I_i, (X_i), (Z_i)$
$Z_4, (X_3X_5), (X_iZ_3, X_2Z_5)$	0011	$I_i, (X_i), (Z_i)$
$Z_5, (X_iX_4), (X_2Z_4, Z_iX_3)$	0001	$I_i, (X_i), (Z_i)$

**Table 6a** : Double errors corrected as  $X$  or  $Z$  single errors having same syndrome. The ancillas are designed by " $a_j$ " and the to be protected qubit by " $i$ " with  $i = 1$  and  $j = 1, 2, 3, 4$ .

Error	S	E <sub>i</sub>
$Y_i, (X_3X_4, Z_2Z_5)$	1101	$I_i, (Y_i)$
$Y_2, (X_4X_5, Z_iZ_3)$	1110	$I_i, (Y_i)$
$Y_3, (X_iX_5, Z_2Z_4)$	1111	$I_i, (Y_i)$
$Y_4, (X_iX_2, Z_3Z_5)$	0111	$I_i, (Y_i)$
$Y_5, (X_2X_3, Z_iZ_4)$	1011	$I_i, (Y_i)$

**Table 6b** : Double errors  $X_kX_l$  and  $Z_kZ_l$  corrected as  $Y$  errors having same syndrome.

## 7 Fidelity

The fidelity  $F(\sigma, \rho)$  is one of the mathematical quantities which permits to know how close are two quantum states represented by the density matrix  $\sigma$  and  $\rho$ , by measuring a distance between them [1] :

$$F(\sigma, \rho) = \left| \text{Tr}(\sqrt{\sqrt{\sigma}\rho\sqrt{\sigma}}) \right|^2 \quad (19)$$

In the case of a pure state  $\sigma = |\Psi\rangle\langle\Psi|$  and an arbitrary state  $\rho$ , the fidelity is the overlap between them [1] :

$$F(|\Psi\rangle, \rho) = \langle\Psi|\rho|\Psi\rangle \quad (20)$$

In this work we measure the overlap between the useful qubit input state  $\sigma = |\Psi\rangle\langle\Psi|$  and the output state  $\rho = |\Psi^E\rangle\langle\Psi^E|$  affected by channel error  $E$  and obtained after decoding. Then, the fidelity is function of the angles  $(\theta, \phi)$  in the Bloch sphere and the average fidelity is :

$$F_a = (1/4\pi) \iint F(\theta, \phi) \sin(\theta) d\theta d\phi, \text{ with } 0 \leq \theta \leq \pi \text{ and } 0 \leq \phi \leq 2\pi \quad (21)$$

Consider any double channel error  $E_k E_l$  occurring on the logical qubit "ia" (useful qubit "i" protected by ancillas "a") during the transmission through a depolarizing channel and corrected as the single error with similar syndrome. Suppose  $P$  the probability that any single channel error  $E_i$  occurs on qubit "i". Then, the global density matrix of the system received is :

$$\rho_{ia}^E = (1-P)^2 \rho_{ia} + P(1-P)(\rho_{ia}^{E_k} + \rho_{ia}^{E_l}) + P^2 \rho_{ia}^{E_k E_l} \quad (22)$$

With  $\rho_{ia}$ ,  $\rho_{ia}^{E_k}$ ,  $\rho_{ia}^{E_l}$  and  $\rho_{ia}^{E_k E_l}$  the density matrix respectively unaffected and affected by  $E_k$ ,  $E_l$  and  $E_k E_l$  errors.

After decoding and suppressing the ancillas we obtain the matrix density of the useful qubit:

$$\rho_i^E = (1-P)^2 \rho_i + P(1-P)(\rho_i + \rho_i) + P^2 \rho_i^{E_i} \quad (23)$$

We note that the single errors  $E_k$  and  $E_l$  are recovered by the three codes and only the double error  $E_k E_l$  will affect the protected qubits by error  $E_i = I_i, X_i, Y_i$  or  $Z_i$ . Then we obtain :

$$\rho_i^E = (1-P^2) \rho_i + P^2 \rho_i^{E_i} \quad (24)$$

We multiply by the initial state  $|\Psi_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle = \cos(\theta/2) |0\rangle + e^{i\phi} \sin(\theta/2) |1\rangle$  and obtain:

$$\langle \Psi_i | \rho_i^E | \Psi_i \rangle = (1-P^2) \langle \Psi_i | \rho_i | \Psi_i \rangle + P^2 \langle \Psi_i | \rho_i^{E_i} | \Psi_i \rangle \quad (25)$$

$$\text{With } \langle \Psi_i | \rho_i | \Psi_i \rangle = 1 \text{ and } F^{E_i}(\theta, \phi) = \langle \Psi_i | \rho_i^{E_i} | \Psi_i \rangle \quad (26)$$

The fidelity (overlap)  $F(P, \theta, \phi) = \langle \Psi_i | \rho_i^E | \Psi_i \rangle$  is then :

$$F(P, \theta, \phi) = (1-P^2) + P^2 F^{E_i}(\theta, \phi) \quad (27)$$

Finally we obtain the average fidelity :

$$F_a(P) = \left(\frac{1}{4\pi}\right) \iint F(P, \theta, \phi) \sin(\theta) d\theta d\phi \quad (0 \leq \theta \leq \pi, 0 \leq \phi \leq 2\pi) \quad (28)$$

$$F_a(P) = (1-P^2) + P^2 F_a^{E_i} = 1 + P^2 (F_a^{E_i} - 1) \quad (29)$$

Table 8 gives the fidelity and average fidelity for errors  $E_i$  occurring on the useful qubit "i" with probability  $P=1$ .

$E_i$	$F^{E_i}(\theta, \phi)$	$F_a$
$X_i$	$ \sin(\theta)\cos\phi ^2$	$1/3$
$Y_i$	$ \sin(\theta)\sin\phi ^2$	$1/3$
$Z_i$	$\cos^2(\theta)$	$1/3$

**Table 7 :** Fidelity  $F(\theta, \phi) = \left| \text{Tr}(\sqrt{\sqrt{\sigma_i}\rho_i^E\sqrt{\sigma_i}}) \right|^2 = \langle \Psi_i | (|\Psi_i^E\rangle\langle\Psi_i^E|) | \Psi_i \rangle$  versus errors on the useful qubits "i".

## 8 Comparison among the three codes

Let us now compare the three codes by computing the average fidelity when double errors occur in a depolarizing channel. For this purpose, we define for any code  $C_n$  an average value  $f_n$  for the parameter  $F_a$  depicted in table 7 :

$$f_n = \frac{[x_n \times 1 + (N_n - x_n) \times \frac{1}{3}]}{N_n} \quad (30)$$

With  $x_n$  the number of double errors letting the useful qubit unaffected and  $N_n$  their total number. From tables 5, 6 and 7 we obtain for the three codes :

$$f_5 = \frac{1}{3} = \frac{27}{81} \quad ; \quad f_7 = \frac{53}{81} \quad ; \quad f_9 = \frac{120}{144} = \frac{5}{6} \quad (31)$$

If the useful qubit "i" is sent without protection through a depolarizing channel it will be affected by error  $E_i = X_i, Y_i$  or  $Z_i$  error, then the affected matrix density received is :

$$\rho_i^E = (1 - P)\rho_i + P\rho_i^{E_i} \quad (32)$$

Then the average fidelity:

$$F_a(P) = (1 - P) + PF_a^{E_i} = 1 - \frac{2}{3}P \quad (33)$$

With  $F_a^{E_i} = \frac{1}{3}$  for  $E_i = X_i, Y_i$  or  $Z_i$

The table 8 depicts the expression of average fidelity for each code.

Code	$C_0$	$C_n$	$C_5$	$C_7$	$C_9$
$F_a(P)$	$1 - \frac{2}{3}P$	$1 + P^2(f_n - 1)$	$1 - \frac{2}{3}P^2$	$1 - \frac{28}{81}P^2$	$1 - \frac{1}{6}P^2$

**Table 8 :** The average fidelity calculated for the n-qubits codes, when double errors occur in a depolarizing channel. The symbols  $C_0$  and  $C_n$  represent respectively no protection and an  $(n, k = 1)$  code.

The figure 5 shows that the nine qubits code gives the best fidelity average fidelity  $F_a(P)$  for all values of  $P$ , followed by the seven then the five qubits codes.

**Figure 5 :** The average fidelity with and without correction.

## 5. Conclusion

This work is the first attempt to use Feynman program to simulate quantum correction. Our results show that this program allows to manipulate easily the qubits states and the operators acting on them. The running time is low when we act on a single qubit but increases with the number of qubits on which the gates act. We have verified that the output are correct for an X, Z or Y input error on the first qubit. Also, the simulation allows us to conclude that the nine qubits code gives the best average fidelity, followed by the seven then the five qubits code, regardless the depolarizing channel error probability  $P$ . We have considered triple errors and more very unlikely and then with negligible effect on the obtained results.

## Acknowledgement

We would like to thank very much Mrs. S.Fritzsche and T.Radtke for providing us with the version 4 (2008) of Feynman Program.



## 6. References

- [1] M.A. Nielsen, I.L Chuang: 'Quantum computation and quantum information', Cambridge University Press, UK, 2000.
- [2] Daniel Gottesman, "An Introduction to Quantum Error Correction" , Proceeding of Symposium in Applied Mathematics. 2000.
- [3] S. J.Lomonaco, Jr. , "A Rosetta Stone for Quantum Mechanics with an Introduction to Quantum Computation", Proceeding of Symposis in Appl Math. 2000.
- [4] P.W.Shor, Scheme for reducing decoherence in quantum computer memory. Phys.Rev. A, 52(4):R2493-R2496, Oct 1995.
- [5] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems', I. Quantum gates and registers, Computer Physics Communications, 2005.
- [6] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems', II. Quantum states, Computer Physics Communications, 2006.
- [7] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems', III. Quantum operations, Computer Physics Communications, 2007.
- [8] T.Radtke, S.Fritzsche: 'Simulation of n-qubits quantum systems',IV. Parametrizations of quantum states, Computer Physics Communications, 2008.
- [9] A. M. Steane. Multiple particle interference and quantum error correction. Proc. R. Soc. Lond. A, 452:2551–2576, 1996. quant-ph/9601029.
- [10] Austin G. Fowler, "Constructing arbitrary Steane code single logical qubit fault-tolerant gates", AarXiv:quant-ph/0411206v2, 20 Dec 2010.
- [11] R.Laflamme,1 C.Miquel,1,2 J.Pablo Paz,1,2 and Wojciech H.Zurek1, "Perfect..Code", Physic Review Letters, Volume 77, Number 1, july 1996.

## Appendix : Simulations

### A. Notations

**Shor** := proc (Co, Cx, Cy, Cz, n, k): the procedure called "Shor" takes as input the amplitude probability Co, Cx, Cy, Cz for respectively, no error, X, Y and Z error on the  $n^{\text{th}}$  qubit. The input  $k=0$  is chosen if we want to detect and correct error before decoding. This choice is useful if we want to use again the ancillas in their original state  $|0\rangle$ . We choose  $k=1$  if we suppress the ancillas at the end, after measuring their final states to know the output error. In fact, this last choose is necessary if an arbitrary error occurs because the correction procedure does not work in this case.

**Psi0** := Feynman\_evaluate("Kronecker product", <a, b>, <1, 0>, <1, 0>): Tensor product of the three first qubit states <a, b>, <1, 0> and <1, 0>.

**H**:= Feynman\_quantum\_operator("H"): matrix for the Hadamard gate.

**X**:= Feynman\_quantum\_operator("X"): matrix for the bit flip X gate.

**CN12** := Feynman\_quantum\_operator(3, "cnot", [1, 2]): matrix for the cnot gate with qubit 2 as target and 1 as controller.

**Ha** := Feynman\_quantum\_operator("HHH"): Three matrix H tensor product.

**Id**:=Feynman\_quantum\_operator("IIIIIIII"): The matrix identity I application on all the qubits states.

**A**:= Feynman\_evaluate("Kronecker power", <1, 0>, 6): Six identical states <1, 0> tensor product.

**Psioc**:= Feynman\_evaluate("Kronecker product", Psiob, A): Application of the operator A on the state Psiob.

**P1**:= Feynman\_quantum\_operator("permute", [1, 4, 5, 2, 6, 7, 3, 8, 9]): Change the order of the qubits so that, for example, the qubit 4 will be in the second position.

**Xa**:= Feynman\_quantum\_operator(9, "X", [1]): The X gate application on the first qubit in a nine qubits system.

**T231**:= Feynman\_quantum\_operator(9, "ccn", [2, 3, 1]): The Toffoli gate matrix with qubit 2 and 3 as controllers and 1 as target in the nine qubits system.

**Hb**:= Feynman\_quantum\_operator(9, "HHH", [1, 2, 3]): Application of H on the first, second and third qubits and the identity matrix I on the six other ones.

**Psi33**:= simplify(Feynman\_print(Psi3)): Simplify and display the state Psi3 in the Dirac notation.

### B. Bit flip code

```
>with(Feynman): with(LinearAlgebra): with(Typesetting): Digits := 20:
Bitflip:=proc (n)
local Psi0, CN12,CN13,X,T321,E,Psi1,Psi2, Psi3,Psi33:
# Initial state
Psi0:=Feynman_evaluate("Kronecker product",<a, b>, <1, 0>, <1, 0>):
# Coding
```

```

CN12:=Feynman_quantum_operator(3, "cnot", [1, 2]):
CN13:=Feynman_quantum_operator(3, "cnot", [1, 3]):
Psi11:=CN12.Psio:
Psi1:=CN13.Psi11:
# Error
X := Feynman_quantum_operator("X"):
if n = 0 then E := Feynman_quantum_operator("III");
print("No error"); end if:
if n = 1 then E:=Feynman_quantum_operator(3,"X",[1]);
print("Error X on first qubit"); end if:
if n=2 then E:= Feynman_quantum_operator(3, X",[2]);
print("Error X on second qubit"); end if:
if n=3 then E:=Feynman_quantum_operator(3, "X", [3]);
print("Error X on third qubit");end if:
Psi2:= E.Psi1:
# Decoding
T321:= Feynman_quantum_operator(3,"ccn",[3,2,1]):
Psi31:=CN13.Psi2:
Psi32:=CN12.Psi31:
Psi3:=T321.Psi32:
Psi33:=Feynman_print(Psi3);print(Psi3 = Psi33);
end proc;

```

### C. Phase flip code

```

>with(Feynman): with(LinearAlgebra): with(Typesetting): Digits:=20:
Phaseflip:=proc(n)
local Psio,Psi1,Psi2,Psi3,Psi33,T,H,Ha,CN12,CN13,Hb,Z,Za:
# Initial state
Psio:=Feynman_evaluate("Kronecker product",<a,b>,<1,0>,<1,0>):
# Encoding
H:=Feynman_quantum_operator("H"):
Ha:=Feynman_quantum_operator(3,"H",[1]):
CN12:=Feynman_quantum_operator(3,"cnot",[1,2]):
CN13:=Feynman_quantum_operator(3,"cnot",[1,3]):
Hb:=Feynman_quantum_operator("HHH"):
Psi11:=Ha.Psio:
Psi12:=CN12.Psi11:
Psi13:=CN13.Psi12:
Psi1:=Hb.Psi13:
# Error
Z:=Feynman_quantum_operator("Z"):
if n=0 then Za:=Feynman_quantum_operator("III"):
Psi2:= Za.Psi1: print("No error"); fi:
if n=1 then Za:=Feynman_quantum_operator(3,"Z",[1]):

```

```

Psi2:=Za.Psi1: print("Phase flip on first qubit"); fi:
if n=2 then Za:=Feynman_quantum_operator(3,"Z",[2]):
Psi2:=Za.Psi1: print("Phase flip on second qubit"); fi:
if n=3 then Za:=Feynman_quantum_operator(3,"Z",[3]):
Psi2:=Za.Psi1: print("Phase flip on third qubit");fi:
# Decoding
T:=Feynman_quantum_operator(3,"ccn",[3,2,1]):
Psi3:=Ha.T.CN12.CN13.Hb.Psi2:
Psi33:=Feynman_print(Psi3): print(Psi3=Psi33);
end proc;

```

## D. Shor code

```

>with(Feynman): with(LinearAlgebra): with(Typesetting): Digits := 20:
Shor:=proc(Co,Cx,Cy,Cz,n,k)
local Psio,H,X,Y,Z,X1,Z1,X2,Z2,X3,Z3,X4,Z4,X5,Z5,X6,Z6,X7,Z7,X8,Z8,X9,
X123,X456,X789,Z9,X16,X49,Id,CN,CN1,Psioa,Ha,Psiob,PsioC,Psiod,CN2,
CN3,CN4,CN5,CN6,CN7,E,Ex,Ey,Ez,Psi1,Psi2,Psi1a,Psi2a,Psi2b,Psi2c,Psi2d,
Psi2dd,Hb,Psi3a,Psi3b,Psi33,T1,T2,T3,T4,A,P1,P2:
# Initial state
Psio:=Feynman_evaluate("Kronecker product",<a,b>,<1,0>,<1,0>):
# Coding
H:=Feynman_quantum_operator("H"):
CN :=Feynman_quantum_operator(3,"cnot",[1,2]):
CN1:=Feynman_quantum_operator(3,"cnot",[1,3]):
Psioa1 :=CN.Psio:
Psioa:=CN1.Psioa1:
Ha:=Feynman_quantum_operator("HHH"):
Psiob:=Ha.Psioa:
A:=Feynman_evaluate("Kronecker power",<1,0>,6):
PsioC:=Feynman_evaluate("Kronecker product",Psiob,A):
P1:=Feynman_quantum_operator("permute",[1,4,5,2,6,7,3,8,9]):
Psiod:=P1.PsioC:
CN2:=Feynman_quantum_operator(9,"cnot",[1,2]):
CN3:=Feynman_quantum_operator(9,"cnot",[1,3]):
CN4:=Feynman_quantum_operator(9,"cnot",[4,5]):
CN5:= Feynman_quantum_operator(9,"cnot",[4,6]):
CN6:=Feynman_quantum_operator(9,"cnot",[7,8]):
CN7:= Feynman_quantum_operator(9,"cnot",[7,9]):
Psi1a1:=CN6.Psiod:
Psi1a2:= CN4.Psi1a1:
Psi1a3:= CN2.Psi1a2:
Psi1a4:=CN7.Psi1a3:
Psi1a5:=CN5.Psi1a4:
Psi1:=CN3.Psi1a5:
# Error operators

```

```

X:= Feynman_quantum_operator("X");
Y:= Feynman_quantum_operator("Y");
Z:=Feynman_quantum_operator("Z");
Id:=Feynman_quantum_operator("IIIIIIII"):
if n=0 then print("No error");
Ex:= Id:Ey:= Id:Ez:= Id:end if;
if n=1 then if Cx=1 then print("X error on first qubit");end if;
if Cy=1 then print("Y error on first qubit");end if;
if Cz=1 then print("Z error on first qubit");end if;
if Cx<1 and Cy<1 and Cz<1 then print("Error on first qubit");end if;
Ex:=Feynman_quantum_operator(9,"X",[1]):
Ey:=-I.Feynman_quantum_operator(9,"Y",[1]):
Ez:=Feynman_quantum_operator(9,"Z",[1]):end if;
if n=2 then if Cx=1 then print("X error on second qubit");end if;
if Cy=1 then print("Y error on second qubit");end if;
if Cz=1 then print("Z error on second qubit");end if;
if Cx<1 and Cy<1 and Cz<1 then print("Error on second qubit");end if;
Ex:=Feynman_quantum_operator(9,"X",[2]):
Ey:=-I.Feynman_quantum_operator(9,"Y",[2]):
Ez:= Feynman_quantum_operator(9,"Z",[2]):end if;
if n=3 then if Cx=1 then print("X error on third qubit");end if;
if Cy=1 then print("Y error on third qubit");end if;
if Cz=1 then print("Z error on third qubit");end if;
if Cx<1 and Cy<1 and Cz<1 then print("Error on third qubit");end if;
Ex:=Feynman_quantum_operator(9,"X",[3]):
Ey:=-I.Feynman_quantum_operator(9,"Y",[3]):
Ez:=Feynman_quantum_operator(9,"Z",[3]):end if;
if n=4 then if Cx=1 then print("X error on forth qubit");end if;
if Cy=1 then print("Y error on forth qubit");end if;
if Cz=1 then print("Z error on forth qubit");end if;
if Cx<1 and Cy<1 and Cz<1 then print("Error on forth qubit");end if;
Ex:= Feynman_quantum_operator(9,"X",[4]):
Ey:=-I.Feynman_quantum_operator(9,"Y",[4]):
Ez:= Feynman_quantum_operator(9,"Z",[4]):end if;
if n=5 then if Cx=1 then print("X error on fifth qubit");end if;
if Cy=1 then print("Y error on fifth qubit");end if;
if Cz=1 then print("Z error on fifth qubit");end if;
if Cx<1 and Cy<1 and Cz<1 then print("Error on fifth qubit");end if;
Ex:= Feynman_quantum_operator(9,"X",[5]):
Ey:=-I.Feynman_quantum_operator(9,"Y",[5]):
Ez:= Feynman_quantum_operator(9,"Z",[5]):end if;
if n=6 then if Cx=1 then print("X error on sixth qubit");end if;
if Cy=1 then print("Y error on sixth qubit");end if;
if Cz=1 then print("Z error on sixth qubit");end if;
if Cx<1 and Cy<1 and Cz<1 then print("Error on sixth qubit");end if;
Ex:= Feynman_quantum_operator(9,"X",[6]):

```

```

Ey:=-I.Feynman_quantum_operator(9,"Y",[6]):
Ez:=Feynman_quantum_operator(9,"Z",[6]):end if:
if n=7 then if Cx = 1 then print("X error on seventh qubit");end if:
if Cy=1 then print("Y error on seventh qubit");end if:
if Cz=1 then print("Z error on seventh qubit");end if:
if Cx<1 and Cy<1 and Cz<1 then print("Error on seventh qubit");end if:
Ex:= Feynman_quantum_operator(9,"X",[7]):
Ey:=-I, Feynman_quantum_operator(9,"Y",[7]):
Ez:= Feynman_quantum_operator(9,"Z",[7]):end if:
if n=8 then if Cx=1 then print("X error on eight qubit");end if:
if Cy=1 then print("Y error on eight qubit");end if:
if Cz=1 then print("Z error on eight qubit");end if:
if Cx<1 and Cy<1 and Cz<1 then print("Error on eight qubit");end if:
Ex:=Feynman_quantum_operator(9,"X",[8]):
Ey:=-I.Feynman_quantum_operator(9,"Y",[8]):
Ez:= Feynman_quantum_operator(9,"Z",[8]):end if:
if n=9 then if Cx=1 then print("X error on ninth qubit");end if:
if Cy=1 then print("Y error on ninth qubit");end if:
if Cz=1 then print("Z error on ninth qubit");end if:
if Cx<1 and Cy<1 and Cz<1 then print("Error on ninth qubit");end if:
Ex:=Feynman_quantum_operator(9,"X",[9]):
Ey:=-I.Feynman_quantum_operator(9,"Y",[9]):
Ez:= Feynman_quantum_operator(9,"Z",[9]):end if:
# Error
E:=Co.Id+Cx.Ex+Cy.Ey+Cz.Ez :
Psi2 :=E.Psi1:
if k = 0 then print("Correction before decoding");
# Gates for error detection and correction
Z1 := Feynman_quantum_operator(9,"Z",[1]):
Z2 := Feynman_quantum_operator(9,"Z",[2]):
Z3 := Feynman_quantum_operator(9,"Z",[3]):
Z4 := Feynman_quantum_operator(9,"Z",[4]):
Z5 := Feynman_quantum_operator(9,"Z",[5]):
Z6 := Feynman_quantum_operator(9,"Z",[6]):
Z7 := Feynman_quantum_operator(9,"Z",[7]):
Z8 := Feynman_quantum_operator(9,"Z",[8]):
Z9 := Feynman_quantum_operator(9,"Z",[9]):
X1:=Feynman_quantum_operator(9,"X",[1]):
X2:=Feynman_quantum_operator(9,"X",[2]):
X3:=Feynman_quantum_operator(9,"X",[3]):
X4 := Feynman_quantum_operator(9,"X",[4]):
X5 := Feynman_quantum_operator(9,"X",[5]):
X6:=Feynman_quantum_operator(9,"X",[6]):
X7:=Feynman_quantum_operator(9,"X",[7]):
X8:=Feynman_quantum_operator(9,"X",[8]):
X9:=Feynman_quantum_operator(9,"X",[9]):

```

```

X123:= Feynman_quantum_operator(9,"XXX",[1,2,3]):
X456:= Feynman_quantum_operator(9,"XXX",[4,5,6]):
X789:= Feynman_quantum_operator(9,"XXX",[7,8,9]):
# Bit flip detection and correction
if evalb(Equal(Z1.Z2.Psi2,Psi2)) = true
and evalb(Equal(Z2.Z3.Psi2,Psi2))= true
and evalb(Equal(Z4.Z5.Psi2,Psi2))= true
and evalb(Equal(Z5.Z6.Psi2,Psi2))= true
and evalb(Equal(Z7.Z8.Psi2,Psi2))=true
and evalb(Equal(Z8.Z9.Psi2,Psi2)) = true then
Psi2a:=Psi2:end if:
if evalb(Equal (Z1.Z2.Psi2,Psi2))=false
and evalb(Equal(Z2.Z3.Psi2,Psi2)) = true then
Psi2a:=X1.Psi2:end if:
if evalb(Equal (Z1.Z2.Psi2,Psi2))=false
and evalb(Equal(Z2.Z3.Psi2,Psi2))=false then
Psi2a:=X2.Psi2: end if:
if evalb(Equal (Z1.Z2), Psi2), Psi2)) = true
and evalb(Equal(Z2, Z3), Psi2)) = false
then Psi2a :=X3.Psi2 end if:
if evalb(Equal(Z4.Z5.Psi2, Psi2)) = false
and evalb(Equal(Z5.Z6.Psi2, Psi2)) = true then
Psi2a :=X4.Psi2:end if:
if evalb(Equal(Z4.Z5.Psi2, Psi2))=false
and evalb(Equal (Z5.Z6.Psi2, Psi2) = false then
Psi2a :=X5.Psi2: end if:
if evalb(Equal(Z4.Z5.Psi2), Psi2)) = true
and evalb(Equal (Z5.Z6.Psi2, Psi2) = false then
Psi2a :=X6.Psi2: end if:
if evalb(Equal(Z7.Z8.Psi2,Psi2))=false
and evalb(Equal(Z8.Z9,Psi2,Psi2))=true then
Psi2a:=X7.Psi2:end if:
if evalb(Equal(Z7.Z8.Psi2,Psi2))=false
and evalb(Equal(Z8.Z9.Psi2,Psi2)) = false then
Psi2a:=X8.Psi2: end if: if evalb(Equal(Z7.Z8.Psi2, Psi2))=true
and evalb(Equal(Z8.Z9.Psi2,Psi2))=false then
Psi2a:=X9.Psi2:end if:
# Phase flip detection and correction
if evalb(Equal(X162.X161.Psi2a,Psi2a))=true
and evalb(Equal(X492.X491.Psi2a,Psi2a))=true then
Psi2b:=Psi2a:end if:
if evalb(Equal(X162.X161.Psi2a,Psi2a))=false
and evalb(Equal(X492.X491.Psi2a, Psi2a)) = true then
Psi2b1:=Z3.Psi2a:
Psi2b2:=Z2.Psi2b1:
Psi2b:=Z1.Psi2b2: end if:

```

```

if evalb(Equal(X162.X161.Psi2a, Psi2a))=false
and evalb(Equal(X492.X491.Psi2a,Psi2a))=false then
Psi2b1:=Z6.Psi2a:
Psi2b2:=Z5.Psi2b1:
Psi2b:=Z4.Psi2b2: end if:
if evalb(Equal(X162.X161.Psi2a, Psi2a))=true
and evalb(Equal(X492.X491.Psi2a,Psi2a))=false then
Psi2b1:=Z9.Psi2a:
Psi2b2:=Z8.Psi2b1:
Psi2b:=Z7.Psi2b2: end if:
end if:
if k=1 then print("Decoding without correction");
Psi2b:=Psi2:end if:
# Decoding
T1:=Feynman_quantum_operator(9,"ccn",[2,3,1]):
T2:=Feynman_quantum_operator(9,"ccn",[5,6,4]):
T3:=Feynman_quantum_operator(9,"ccn",[8,9,7]):
Psi2c1:=CN6.Psi2b:
Psi2c2:=CN4.Psi2c1:
Psi2c:=CN2.Psi2c2:
Psi2d1:=CN7.Psi2c :
Psi2d2:=CN5.Psi2d1 :
Psi2d3:=CN3.Psi2d2 :
Psi2d4:=T3.Psi2d3 :
Psi2d5:=T2.Psi2d4 :
Psi2d=T1.Psi2d5 :
P2:= Feynman_quantum_operator("permute", [1,4,7,2,3,5,6,8,9]):
Psi3a:=P2.Psi2d :
Hb := Feynman_quantum_operator(9,"HHH",[1,2,3]):
Psi3b:=Hb.Psi3a :
T4:=Feynman_quantum_operator(9,"ccn",[2,3,1]):
Psi31:=CN3.Psi3b :Psi32:=CN2.Psi31 :Psi3:=T4.Psi32 :
# Displaying the final state in the Dirac notation
Psi33:=simplify(Feynman_print(Psi3)):
print(Psi3afterdecoding = Psi33);
end proc:

```